Program #5
Due: Wednesday Apr 8th, 2015 at 11:30PM

Instructor	Dr. Stephen Perkins
Office Location	ECSS 4.403
Email Address	stephen.perkins@utdallas.edu
Office Hours	Monday and Wednesday before class (7:30pm - 8:25pm) And by appointment
TAs	Kenneth Joseph Platz (kxp101120@utdallas.edu) Xuming Zhai (xxz124830@utdallas.edu)
TA Office Location	Kenneth Platz Clark Center CN 1.202C Xuming Zhai Clark Center CN 1.202D
TA Office Hours	Kenneth Platz Tuesday/Thursday 12:00pm-2:00pm Xuming Zhai Monday/Wednesday 5:00pm-6:00pm

Purpose

Demonstrate the ability to download, build, install and use 3rd party library code distributed with SCM tools (git), GNU Automake (RudeConfig), and custom build tools (Libcrypt++). Demonstrate the ability to parse command line options. Demonstrate the ability to parse config files. Demonstrate the ability to create a daemon process. Demonstrate the ability to log to a file. Demonstrate the ability to control processes via UNIX signals. Demonstrate the ability to utilize the *inotify* facility of UNIX. Demonstrate the ability to control and manage a full multi-file UNIX project from the command line using all needed tooling.

Assignment

Overview

The goal of this program is to create a UNIX style daemon process that will watch a folder and respond to specific file operations that occur in that folder. The daemon process will read its configuration parameters from a CONFIG file and then operate according to that configuration. The daemon process will also respond to several UNIX signals that will help control its operation.

Once operational, the daemon will act on any files that have been written to the folder that it is observing. When a write is detected, the daemon will immediately create a copy of the just written file and store that copy in a ***.versions*** subfolder. Each copy stored in the ***.versions*** subfolder will have a date stamp appended to the filename.

Usage (Command Line Arguments)

Your code must use the TCLAP library (http://tclap.sourceforge.net) for parsing the command line. You are to accept one optional flag and one optional filename:

-d	Run in daemon mode. If this optional flag is given, the system will fork and run in the background as a daemon. If not provided, the program should run in the console as would any other program.
<config filename>	If this optional filename is given, it is the name of the config file that should be parsed for operational parameters. If the filename is not provided, you should use the default filename *cs3376dirmond.conf*.

The specific usage as returned by TCLAP is:

```
USAGE:

   ./cs3376dirmond  [-d] [--] [--version] [-h] <config filename>

Where:

   -d,  --deaemon
     Run in daemon mode (forks to run as a daemon).

   --,  --ignore_rest
     Ignores the rest of the labeled arguments following this flag.

   --version
     Displays version information and exits.

   -h,  --help
     Displays usage information and exits.

   <config filename>
     The name of the configuration file. Defaults to cs3376dirmond.conf

   cs3376dirmond Directory Monitor Daemon
```

The CONF file

The CONF file (default name of *cs3376dirmond.conf*) is a text file that contains the initial settings for the daemon. The file is grouped into a set of sections, and each section can have zero or more definitions. A definition is an Identifier followed by an = sign and then followed by a value. Specifically:

Format:

Blank lines are ignored.
Lines in the file that start with the # symbol are considered comments.
Section names are on a line by themselves and are enclosed in square brackets.
Definitions take the form: ID = VALUE (each definition is on its own line).

The following definitions should be in your file. They ***must be included*** in the section named: **[Parameters]**

ID	Meaning	Example
Verbose	If true, log more verbose messages.	Verbose=true
LogFile	The name of the LogFile. If not provided, the default is cs3376dirmond.log.	LogFile=cs3376dirmond.log
Password	The password used to encrypt files.	Password=mypassword
NumVersions	The number of file versions to keep for each filename in the .versions folder.	NumVersions=10
WatchDir	The directory that the daemon will monitor.	WatchDir=/people/cs/x/xp12540/testdir

If the CONF file is missing, the **[Parameters]** section is missing, or there are definitions missing from the CONF file, you should log an appropriate error message and exit.

You must use the ***RudeConfig*** library for parsing your CONFIG file. You will need to download, compile, install, and then use the library as part of this project. See:

http://rudeserver.com/config/

The PID file

The daemon will create and remove a special file called *cs3376dirmond.pid*. While the daemon is running, this file should contain the process ID (PID) of the daemon. When the daemon exits or is interrupted, it should remove this file. If the daemon finds this file already present when it is starting up, it should log an error message and exit. Only one copy of a daemon should be running at any given time.

Signals

The daemon must be able to respond to the following UNIX signals:

SIGINT or SIGTERM - The daemon should remove the PID file, close logging, and exit.

SIGHUP - The daemon should re-read its CONF file and reconfigure itself with any new settings. This may entail changing the logging verbosity, the log file name, or the number of versions of a file to keep. You ***should not allow*** changes to the *password* or the *watch directory* while the daemon is running.

.versions

Upon starting, the daemon will create a subfolder in the ***WatchDir*** called *.versions*. This is the folder that will receive copies of modified files. If the folder already exists, then the daemon will use the existing folder.

Time Stamp Format

When modified files are copied to the *.versions* folder, a timestamp will be appended to the name of the file. The timestamp will take the following format:

<Original file name>.<year>.<month>.<date>-<hour>.<minute>.<seconds>

Ex: junk.txt.2015.03.25-21:43:47

inotify

Linux has a facility that will notify programs when the contents of a folder change. This facility is called ***inotify***. You must use the ***inotify*** facility to monitor ***WatchDir*** for *IN_MODIFY* events. These are the events that trigger your daemon to make a time-stamped copy of a file. More information can be found online and in the man page.

Logging

Your daemon should create and open a logfile as specified by the *LogFile* parameter in the CONF file. This file must be closed when the program/daemon exits. You are free to log initial messages to stdout as you get started and read the CONFIG file.

Deliverables

You must submit your homework through ELearning. You must include a tarball that contains your Makefile, .h, .c, .conf, and any other important files. All source files, CONF files, and Makefiles need to have your name, email, and course number commented at the top. As usual, the code must compile and run on cs1.utdallas.edu. You should use the ***-static*** flag when linking. INCLUDE A COPY of your statically linked executable with your tarball.

Hints

- You may use the shell's ***date*** command to generate a date string in the proper format.
- You may use the shell's ***cp*** command to make copies of files.
- You can use the ***stat*** function with error ***EBADF*** to check for file existence.
- You can use the ***kill*** command to stop or signal your running daemon. See the man page.
- Don't forget to add appropriate -I, -L, -l flags to your compiler calls
- The NumVersions CONFIG file setting is not used.

Notes

Your code must be compiled using the **--Wall** compiler flag and must produce no errors/warnings.

You should link with the -static flag.

Make sure you include a copy of your statically linked executable with your tarball.

No late homework is accepted.

Make sure to use the ***ps*** command and check that you are not leaving your daemons running once you log off.

Make sure that you do not run more than one daemon at any given time.

Recommended Project Phases

This project is complex. In order to help with organization, students might consider using these phases as a guideline in how to write the code. Code should also be spread among separate files to keep code clean. Specifically:

Phase 1	• Create Makefile that compiles runs a hello world program
	  ○ should use -I flags with include paths
	  ○ should use -L with library paths
	  ○ should use -l flags with libraries
	• verify backup target working (for backups and program submission)

Phase 2	• download/install tclap (need to add tclap library location to Makefile) • Use tclap to to parse your command line • Create map object that holds values from command line
Phase 3	• download/build/install RudeConfig (need to add RudeConfig library location to Makefile) • Use RudeConfig to parse your config file • Place config file parameters into map object created in Phase 2

Phase 4	• Combine all of the above into a program that can:
	  ○ Run from command line with no -d
	  ○ Run as a daemon (with -d and using *fork*())
	  ○ create .pid file
	  ○ use config file
	  ○ create log file
	  ○ respond to signal
	  ▪ remove pid file and close log file
	  ▪ reread config file

Phase 5	• Use the inotify interface to log events for WatchDir

Phase 6	• Use inotify to implement the versioning to *.versions* folder
	  ○ Generate date-stamped file names
	  ○ copy files to the *.versions* folder.

Here are the files I chose to use. You do not have to use these specific names or grouping. This is for reference only:

cs3376dirmond.conf
inotifyFunctions.cc
loggingFunctions.cc
Makefile
parseCommandLine.cc
parseConfigFile.cc
processControl.cc
program5.cc
program5.h
signalHandling.cc
tarballs (this is a folder that contains the .tar.gz files for the libraries I used: crypto++, rudeconfig, tclap)

The actual libraries and includes that I needed from building the libraries are kept in:
~/libs
~/include

I used appropriate Makefile flags to let the compiler know where to find them.

Extra Credit (25 points)

For extra credit, a student may also elect to add the following features. BOTH features must be available for this to qualify.

1. In addition to the daemon creating time-stamped copies of files in the ***.versions*** subfolder, the daemon will create a ***.secure*** subfolder. Files written to this folder are time-stamped and encrypted using the password as specified by the *Password* parameter in the CONF file.

2. A standalone program must be created that can decrypt a file given a password. The program will accept the encryption password (<password>), the encrypted file (<infile>), and the filename for output (<outfile>) on the command line Its USAGE should be:

   cs3376decrypt <password> <infile> <outfile>

<password>	The password to use for decryption
<infile>	The existing encrypted file
<outfile>	A file that is created and contains the decrypted version of <infile>

You must use the ***Crypto***++ library for encryption and decryption. You will need to download, compile, install, and then use the library as part of the extra credit. See:

http://www.cryptopp.com/

Hints

- See the ***EncryptFile*** and ***DecryptFile*** functions in test.cpp

---

Extra Credit (15 points)

For extra credit, a student may use the *NumVersions* CONFIG file definition to limit the number of copies of a file that the daemon will keep in the *.versions* subfolder. When a new file is added such that the number of copies of that file is greater than *NumVersions value*, the oldest versions of the file are deleted such that there are exactly *NumVersions* of that file available.